printed, faxed, or saved to a PostScript file. Not only has printing become a nonissue in our NeXTSTEP programs, but adding windows to an application for expository text or pictures is trivial. Drag a window from the palette in IB, drag a text object into the window, and start typing. The window will even create scroll bars if the text exceeds the size of the exposed view.

Serious contender

With the exception of a good Pascal environment for teaching our introductory computational-physics course, we have been able to fill every departmental application need either through public domain, shareware, or commercial applications at PC prices. Our current standard application suite includes word processors, symbolic-computation package, equation editor, image editor and drawing tools, and a spreadsheet.

NeXT ought to be considered a serious operating system for computational physics, particularly in small departments with limited computer support. NeXT bundled applications fill many system-administration, communication, and programming needs. Reasonably priced commercial applications exist. Once NeXTSTEP has been set up and is running, it requires less software administrative work, and is far more functional, than single-user operating systems. Our students have given NeXTSTEP the highest compliment—when identical applications are available on two or three different operating systems, they usually choose to work with NeXTSTEP.

This review is based in the author's experience managing and programming a cluster of 14 NeXTstations, as part of a NeXT, DOS-/Windows, Apple network in the Davidson College Physics Department. This department's NeXTSTEP program LASER won an honorable mention in the 1992 *Computers in Physics* software competition.

INTERACTIVE PHYSICS II: A PHYSICS SIMULATION LABORATORY FOR THE MACINTOSH

A. John Mallinckrodt

Interactive Physics II system requirements:
Macintosh computer with System 6.0.5 or later, hard disk, 1500-kbytes free RAM for black and white or 2500-kbytes free RAM for 8-bit color.
Available from: Knowledge Revolution, 15 Brush Place, San Francisco, CA 94103. Order no. (800) 766-6615; Tel: (415) 553-8153; Fax: (415) 553-8012. List price: IP II $399; IP II student version (no technical support or upgrade rights) $99; Fun Physics (the original Interactive Physics) $99; educational discounts available for lab packs.

John Mallinckrodt is Associate Professor of Physics at California State Polytechnic University, Pomona, CA 91768. He likes to call himself a generalist; he is, however, particularly interested in the use of computer-based simulations in physics education. E-mail: ajmallinckro@csupomona.edu.

Not quite four years ago, at the summer meeting of the American Association of Physics Teachers in San Luis Obispo, CA, attendees were buzzing about a remarkable new Macintosh application being demonstrated by its young author, David Baszucki. By now virtually every physicist with an interest in educational software is aware of that product—Interactive Physics (IP)—and its unique ability to simulate the dynamics of two-dimensional mechanical systems that are literally drawn using the mouse. The original package, reviewed in the May/June 1991 issue of *Computers in Physics*,[1] has probably had more impact upon the use of computers in primary physics instruction than any other single application, with the possible exception of spreadsheets. Last summer Knowledge Revolution unveiled the greatly enhanced Interactive Physics II (IP II). In this review I shall discuss only a few of the more significant and advanced features making their debut in IP II.

While IP II looks and works very much like its predecessor, and runs experiments written for IP, the underlying "engine" and data structures have been rewritten in order to build a more flexible and reliable base for future improvements. IP II also introduces an impressive array of new features:

- User-definable forces that take the form of two-body or field interactions. (IP II includes predefined forces simulating planetary and terrestrial gravity, electrostatic and magnetic fields, and air resistance.)
- Customizable integration algorithms (see below).
- Actuators and motors that can exert user-defined forces or enforce

user-defined relative positions, velocities, and accelerations.

- A large number of new constraints including rotational springs and dashpots, rods, separators, pivots, keyed or unkeyed slot joints, and pulley systems. (Note: Pulley systems consist of ropes constrained to pass without friction through one or more fixed points.)
- "Controls" that allow the user to change parameters of an experiment during a run and without the need to invoke dialog boxes.
- User-definable graphs and meters.
- Greatly enhanced control over the properties and appearances of objects, including the ability to import pictures (from painting and drawing programs) and, optionally, to attach them to objects (see Fig. 1).
- An improved algorithm for handling frictional contact between objects, which models both static and kinetic friction.
- The ability to view experiments from reference frames—inertial or otherwise—attached to any object or the system center of mass.
- "Player documents" that shield users from the program's full complexity and give them access to a limited number of experimental parameters via "menu buttons" and "controls" (see below).
- Quicktime output of experimental runs for more rapid playback.

But by far the most important addition, and the one that underlies and enhances the flexibility of many of those already mentioned, is a new formula language that allows users to access simulation parameters and to use the values of those parameters to control others.

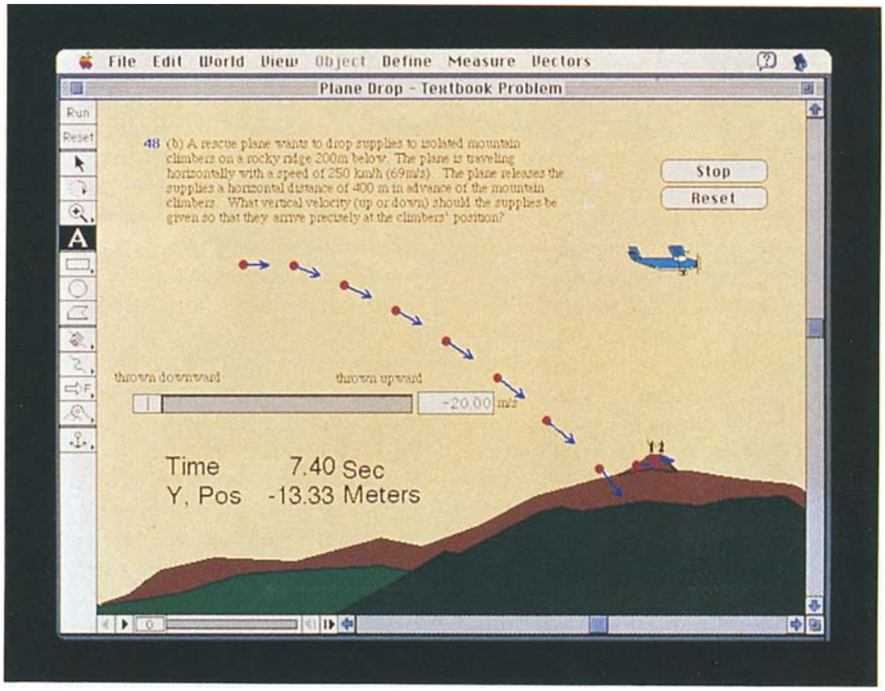


Fig. 1: Frame from an IP II simulation shows the use of imported graphics, controls, menu buttons, formatted text, and meters.

Formulas in IP II

The properties and appearance of objects in IP II are specified in the text fields of a pair of floating windows; for example, one can specify the initial velocity of a body, the torsion constant of a rotational spring, or the variable(s) to be plotted in a graph. Much of the power of IP II, however, derives from the user's ability to substitute formulas for these values in a manner analogous to the use of formulas in spreadsheets.

The IP II formula language assigns to every object an identifier, and to most identifiers a variety of fields that contain information about the object's properties and current dynamic state. Through these the user has access to the values of most simulation parameters, including constraint forces and lengths, meter values, control outputs, friction and normal forces, and the current frame number. One combines or operates on these values with logical and/or mathematical functions to form formulas that control the properties of other objects. Thus, it is a relatively simple matter to create a meter that measures, for instance, the angular momentum of a body relative to some point, moving or not, simply by inserting the proper formula into the field that describes the output of a meter. In a more advanced application, one can control the strength of a magnetic field according to what time it is, according to the location of an object, or even, if one is so perversely inclined, according to the tension in a pulley system. Indeed, if one cares to—or is simply careless—one can quite easily undermine the preprogrammed intention of the IP II engine, which is, of course, to simulate a world governed by Newton's laws.

There are a few inconsistencies in the syntax of the formula language, some confusion and errors in the manual descriptions of functions, and a few mistakes in the formulas that describe the default measurements of some meters. For example, some forces treat the x and y components of a vector as being along and perpendicular to the line between two objects instead of horizontal and vertical. This is often desirable, but it is not always obvious which rule applies, and the manual even gets it wrong in at least one case—that of two-body constraint forces. I suspect that it is at least partially because of this confusion that the default formulas in meters measuring the gravitational or electrostatic potential energy of two bodies are wrong and, as a result, give values that depend upon the direction from one body to the other.

Integration algorithms

At the heart of IP II is the "engine" that carries out the discrete integration of Newton's second law. IP II allows the user to modify the integration algorithm in several ways: One has the choice of three basic methods—Euler, fourth-order Runge-Kutta, and predictor-corrector. One can choose fixed or variable simulation time steps and can also specify a longer time step between screen updates than is used for the simulation itself. (This is a handy feature for increasing the accuracy of the simulation without unnecessarily increasing the graphics burden on the processor.) One can ask the engine to produce warnings when it detects high velocities and/or accelerations or constraints that are redundant or inconsistent. Finally, there is an important but not too well-explained option that pertains to a fundamental difficulty any discrete integrator will have dealing with collisions, especially when those collisions involve multiple bodies, frictional contact, and partial elasticity. Because the handling of such collisions involves a number of somewhat ad hoc operations, neither option here is wholly satisfactory. In general, the user is well advised to view the detailed results of collision intensive experiments with appropriate skepticism.

I tested the integration algorithms with the following experiment: A 1.0-kg body is attached to a damper that applies a force given by $-kv^3$ with $k = 1\ \mathrm{N/(m/s)^3}$. The body is given an initial velocity of 10 m/s. This problem is somewhat challenging to simulate because the retarding force is initially so large (1000 N) and decreases so quickly. Note, for instance, that using Euler's method with a fixed time step of more than 0.01 s will yield a retarding impulse in the first time step that is greater than the initial momentum of the body. As a result, the body's direction of motion will be immediately reversed—an unphysical result of the overly large time step. Using a smaller time step would seem to be necessary to obtain reasonable results. On the other hand, as the velocity decreases, the per-step impulse falls rapidly and a smaller time step is computationally inefficient. Problems like this benefit greatly from the use of variable-time-step methods.

The table shows the results of the six integration methods provided by IP II applied to this problem. For a range of basic step sizes, the table shows the time (in seconds) that it takes the body to move 2.0 m, as measured both in simulation time (for which an analytical calculation yields the result 2.20 s) and in real time on a Mac IIci running in 16-color mode. As you can see, for this problem, the variable-time-step predictor-corrector method was the stand-out champion, giving accurate results for any time step up to the maximum used. As expected, the fixed-time-step Euler method fails for time steps of 0.01 s or more.

As another test of the algorithms I created a simulation of a satellite orbiting a planet in a highly eccentric orbit. This kind of simulation often causes problems because of the large spatial variations in the force and the high velocities that occur near the point of closest approach. I measured the orbital period, the percentage change in energy during the first orbit, and the computation time required to complete the orbit. In this case the variable-time-step Runge-Kutta method emerged as the best choice, with the predictor-corrector method the big loser. Interestingly, the Euler method tended to give the

Table. A comparison of simulation times and real times observed using a test experiment as described in the text. Each row gives the results of a different integration method ("PC" = predictor-corrector, "RK" = fourth-order Runge-Kutta, "var" = variable time step, "fix" = fixed time step), and the columns indicate the basic time steps used. Each entry gives "simulation time/real time" (both in seconds) on a Mac IIci.

	0.2 s	0.1 s	0.05 s	0.02 s	0.01 s	0.005 s	0.002 s
Euler var	2.80/3	2.60/5	2.50/7	2.40/15	2.360/26	2.300/51	2.238/122
Euler fix	—	—	—	—	—	2.435/50	2.276/108
PC var	2.20/3*	2.20/5	2.20/7	2.20/15	2.19/29	2.195/57	2.200/140
PC fix	—	—	—	0.64/6	2.24/24	2.185/44	2.198/111
RK var	—	—	2.25/9	2.20/20	2.21/37	2.205/73	2.202/182
RK fix	—	—	—	2.48/16	2.26/27	2.205/52	2.202/142

* Had to ignore a high-force warning.
— Simulation failed with these parameters.

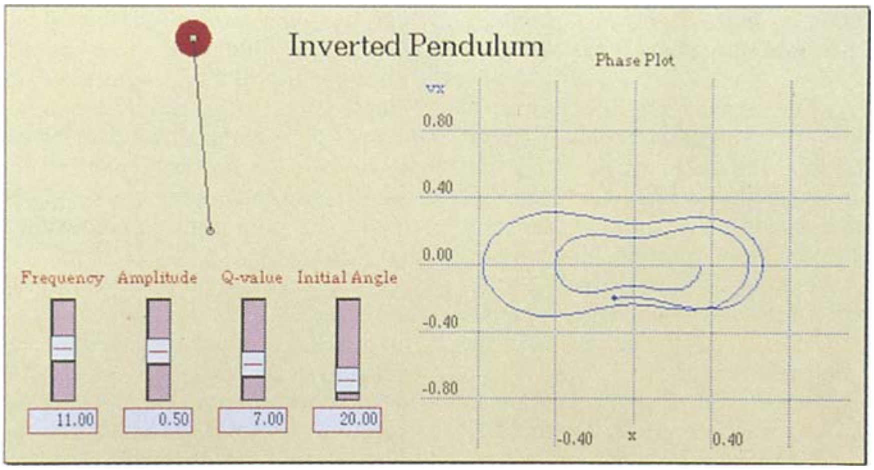


Fig. 2: Simulation of the inverted pendulum, from which this frame was taken, exercises many of the tools that have been introduced in IP II.

smallest cumulative energy change, even though it did vary during the orbit by as much as 60%!

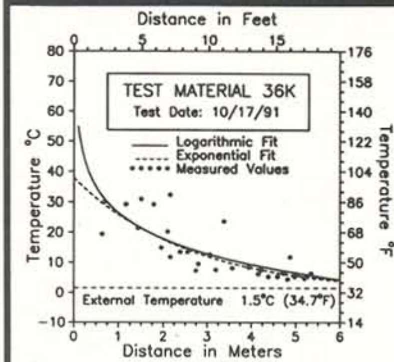
An example

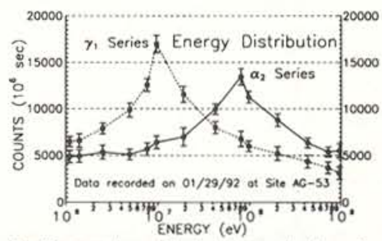
The inverted pendulum[2] is a nonlinear system that exhibits both counterintuitive dynamic stability and chaotic dynamics. Fig. 2 shows a frame from my IP II simulation of the system, which uses many of the new tools in IP II. The rod is connected by a pivot at its lower end to an invisible object whose y coordinate has been specified (using the formula language) as a sinusoidal function of time; the formula references the output of the "Amplitude" control. The initial x and y positions of the circular body attached to the upper end of the rod are determined by formulas that reference the "Initial Angle" control. The circle is also subject to gravity and a linear velocity-dependent drag force that are both implemented using a custom force field. The formulas that define that field reference the outputs of the other two controls. (Note: In order to preserve the accuracy of the chosen time step, the "Frequency" of the oscillator was used to renormalize the gravitation and drag forces.) To create the phase plot, references to the x components of the circle's position and velocity were inserted into the fields that specify the x and y coordinates of the graph. The simulation took about half an hour to create and debug, and reproduces many specific behaviors discussed in the previously referenced article.

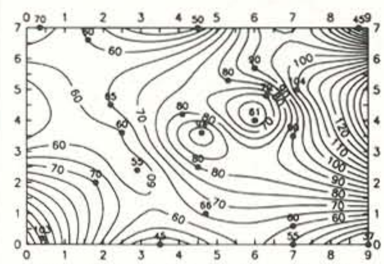
Player documents and course ancillaries

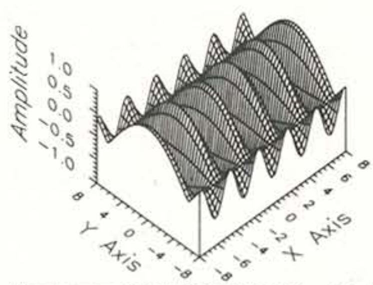
IP II has two modes of operation. When launched it enters "Edit Mode" by default, giving the user full access to all editing tools and menu items. However, experiments can be created and then saved in "Player Mode." Player mode documents shield the user from all commands except those that run, stop, and reset the experiment, and any others that the creator chooses to allow in the form of controls and/or menu buttons. This facility encourages the creation of exercises that focus the attention of the student on the effects of varying specific parameters in a given situation. Knowledge Revolution is promoting the use of these Player Mode documents in hopes of cultivating a lively aftermarket in the exchange of high-quality simulations produced by faculty for use in the classroom and for class assignments. Saunders College Publishing is using Player Mode documents to support its offerings in introductory physics.

In a related vein, Knowledge Revolution has produced the Interactive Physics Player, and has given Prentice-Hall exclusive rights to its use. Course ancillaries, in the form of the IP Player and Player files keyed to specific examples and problems, are available to students using Prentice-Hall's introductory physics texts. It









should be noted that the IP Player is designed to work only with the simulations that come with the Prentice-Hall textbooks, and not with files created by IP II.

Final words

With all of its new features, IP II is slower than its predecessor; on identical simulations, I found that it typically ran at about half the speed of IP. On the other hand, a judicious choice of integration algorithm and step size can counter, and even reverse, this effect without sacrificing accuracy. Furthermore, the new version extracts smaller relative penalties for running in 16-color mode rather than black and white. And finally, it is robust; despite some pretty severe torture at my hands, the program locked up only once.

It is worth noting here that, by the time this review is published, Knowledge Revolution intends to have released a "professional" version of IP that makes use of the math coprocessor, and sports additional features and improved accuracy. There are also plans for a Windows version.

This review is the product of several months of fairly detailed probing into the behavior of IP II and numerous discussions of my findings with David and Greg Baszucki at Knowledge Revolution. Early in my investigations, Knowledge Revolution released a maintenance upgrade for IP II that addressed several of the problems that I and others had discovered. The free distribution of this upgrade to registered users of IP II demonstrates the company's commitment to eliminating the inevitable bugs that crop up in a major revision. That job, however, is not over. There are still a number of annoyances, peculiarities, and bugs—mostly minor, but a few more substantial. They tend to be hidden in the more advanced areas of the program, where many users may never encounter them. I have sent my own detailed list to Knowledge Revolution and, if the past is any indication, most will be addressed in a future maintenance upgrade.

Let there be no doubt about my affection for IP II. It is a flexible and fascinating product that offers a significantly expanded array of features over its predecessor. Despite its occasional idiosyncrasies, I find it to be an efficient tool for preparing and presenting simulations that are effective and readily modified in the classroom; for testing, sharpening, and often correcting one's intuition; and simply for casual explorations.

References

1. A. John Mallinckrodt, Computers in Physics **5**, 349 (1991).
2. See, for example, Blackburn *et al.*, Am. J. Phys. **60**, 903 (1992). A numerical simulation is used to explore the dynamics of this nonlinear system.